IDA

INSTITUTE FOR DEFENSE ANALYSES

Cryptography for a High-Assurance Web-Based Enterprise

William R. Simpson, *Project Leader*

Coimbatore Chandersekaran

October 2013
IDA Non-Standard
NS D-5090
Log: H 2013-001837
Copy

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

About This Publication

This work was conducted by the Institute for Defense Analyses (IDA) under contract DASW01-04-C-0003, Task BC-5-2283, “Architecture, Design of Services for Air Force Wide Distributed Systems,” for USAF HQ USAF SAF/CIO A6. The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

Copyright Notice

4850 Mark Center Drive, Alexandria, Virginia 22311-1882 • (703) 845-2000.

Cryptography for a High-Assurance Web-Based Enterprise

Coimbatore Chandersekaran, William R. Simpson

Abstract— **Each web service and each infrastructure service has a need for symmetric and asymmetric encryption, as well as signature processing and other cryptographic processes. This profile does not include electrical requirements for Multi-Level Systems (MLS). A number of specialized cryptographic functions have been developed for hardware and network operations. Their use is appropriate for network level operations. For purposes of this paper, the discussion is limited to NSA Type 3 for IP enabled communications and similar algorithms. Cryptography is used by most of the services in an enterprise. Asymmetric encryption is performed in suitably security hardened stores and symmetric encryption is performed in most bi-lateral operations. Signatures for integrity and trust use are pervasive. Key management is required throughout the enterprise. The crypto services may be used through all of the *Open Systems Interconnection (*OSI) *model* layers, however, this document concentrates on layers 4 and above.**

Keywords- **cryptogrophy; Public Key Infrastructure; Electronic Signature; Web services; Transport Layer Security; Cryptographic Hash.**

I. Introduction

Cryptographic services are pervasive in a high-assurance enterprise. Some terminology that is discussed follows.

- *Plaintext* – the material that is to undergo cryptographic operations.
- *Key* – The unique mathematical parameter used in the cryptographic operations to generate a *ciphertext.*
- *Cipher* – The algorithm that is used to apply a key and a cryptographic operation.
- *Encryption* – The act of applying the cipher to the *key* and *plaintext,* resulting in *ciphertext.*
- *Ciphertext* – the results of encryption of a *plaintext.*
- *Decryption* – The act of applying the cipher to the key and the *ciphertext,* resulting in *plaintext.*

Cryptographic services and functions need to be standardized to ensure interoperability. These services provide security protection for data and messages and support security properties such as confidentiality, integrity, non-repudiation, authentication and authorization. Functions to sign, validate, and timestamp messages are provided via mechanisms approved for enterprise usage. Authentication and signature cryptographic functions employed in the enterprise are based on the use of the Public Key Infrastructure (PKI) public/private key pairs. The strength of the protection mechanisms for these functions relies on the private key portion remaining secret. The private keys are kept in secure storage, such as on a Hardware Identification Card (HIC). For services and devices, private keys are kept in a secure store.

1. *Cryptographic Functions in WS Security* - Protection for service calls follow the WS Security standards [2] for protecting messages. Several supporting services are required; Extensible Markup Language XML Canonicalization [1c], Hashing function. Secure Hash Algorithm (SHA) -512 [3f], and strong random number generation (*Strong implies uniform distribution of values over the appropriate range).*
2. *Other Cryptographic services* - Java provides many cryptographic services through the Java Cryptography Architecture (JCA) framework. The libraries used to do cryptographic operation must be certified [3d].
3. *Data at Rest* - Data at Rest may be encrypted to protect the confidentiality and integrity of the data. If encrypted, the keys are derived from the primary credentials of the owner of the data.
4. *Data in Motion* - All data in motion are encrypted and transported using Transport Layer Security (TLS) with mutual authentication [5a]**.** End-to-end encryption is used for all active entity communications.

This paper is divided into the following sub-paragraphs for the handling of dependent functions; Cryptographic Keys, Encryption, Decryption, Hash Function, and Signatures

II. Cryptographic Keys and Key Management

Each active entity has an asymmetric key pair issued as part of their Enterprise X.509 [5b] PKI certificate. These keys are RSA 2048 bit asymmetric keys. Symmetric keys are generated for Transport Layer Security and other forms of communication because their computation is more efficient.

A. Generating Asymmetric Keys

All asymmetric keys are generated in a security hardened store such as a hardware storage module, or HIC. All PKI certificates use RSA 2048 bit asymmetric key generation. Users are issued HIC through the normal identification issuance process. All other active entities (machines, devices, servers, web services, and other OSI level-5+ software) are issued enterprise X.509 PKI certificates in the following manner. The keys are generated in a security hardened store where the private key is retained. The public key, distinguished name and other pertinent information is provided to the Enterprise authorized certificate issuing agency which reviews the certification request and issues the Enterprise X.509 PKI certificate.

Manuscript received 22 May 2013; revised 16 July 2013. This work was supported in part by the U.S. Secretary of the Air Force and The Institute for Defense Analyses. The publication of this paper does not indicate endorsement by the Department of Defense or IDA, nor should the contents be construed as reflecting the official position of these organizations

Coimbatore Chandersekaran is with the Institute for Defense Analyses.(email: cchander@ida.org).

William R. Simpson is with the Institute for Defense Analyses, 4850 Mark Center Drive, Alexandria, Virginia 22311 USA and is the corresponding author phone: 703-845-6637, FAX: 703-845-6848 (e-mail: rsimpson@ida.org)

RSA Key generation

The keys for the RSA algorithm are generated by choosing two distinct prime numbers p and q. For security purposes, the integers p and q should be chosen at random, and should be of similar bit-length.

Compute n = pq. n is used as the modulus for both the public and private keys (1)

Compute φ(n) = (p – 1)(q – 1), where φ is Euler's totient function. Choose an integer e such that 1 < e < φ(n) and gcd (e,φ(n)) = 1, i.e. e and φ(n) are coprime.

e is released as the public key exponent. (2)

The bit length for the enterprise cryptographic operation is 2048. Determine d = e^{-1} mod φ(n); i.e. d is the multiplicative inverse of e mod φ(n). This is more clearly stated as solve for d given (d*e) mod φ(n) = 1.

d is kept as the private key exponent. (3)

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the private (or decryption) exponent d which must be kept secret.

B. Generating Symmetric Keys

Symmetric keys are developed for a number uses. One of main reasons to do so is the algorithms for encrypt and decrypt are faster using symmetric keys than similar algorithms for asymmetric keys.

1) TLS Mutual Authentication Key Production

TLS Mutual Auth requires that the TLS client-side holds a certificate (all active entities are required to hold Enterprise X.509 certificates). TLS involves three basic phases; peer negotiation for algorithm support, key exchange and authentication , and symmetric cipher encryption and message authentication During the first phase, the client and server negotiate *cipher suites*, which determine the ciphers to be used, the key exchange and authentication algorithms, as well as the Message Authentication Codes (MACs). The key exchange and authentication algorithms are typically public key algorithms. The message authentication codes are made up from cryptographic hash functions using the HashMAC (HMAC) construction for TLS, and a non-standard pseudorandom function for SSL. The key information and certificates necessary for TLS are handled in the form of X.509 certificates, which define required fields and data formats. In order to generate the session keys used for the secure connection, the client encrypts a random number (RN) with the server's public key, and sends the result to the server. Only the server can decrypt it (with its private key: this is the one fact that makes the keys hidden from third parties, since only the server and the client have access to this data. The client knows Public Key and RN, and the server knows Private Key and (after decryption of the client's message) RN. A third party may only know Public Key, unless Private Key has been compromised. From the random number, both parties generate key material for encryption and decryption.

2) Other Key Production

Symmetric keys may be generated for a number of other reasons, including safe store of data during operations and encryption of memory for protection. Keys must be generated using a strong Random Number Generator and must be at least 512 bits in length. Each generation is coupled with a crypto-algorithm that is used for the encryption and decryption of material. The enterprise uses AES for its encryption algorithm. A cryptographic module of any web service may optionally implement an internal key generation function. The module implements a FIPS approved key generation algorithm. When a random number generator is used in the key generation process, all values are generated randomly or pseudo-randomly such that all possible combinations of bits and all possible values are equally likely to be generated. A seed key, if used, is entered in the same manner as cryptographic keys. Intermediate key generation states and values are not accessible outside of the module in *plaintext* or otherwise unprotected form [3e].

C. Store Keys

All asymmetric keys are generated in a security hardened store such as a hardware storage module, HIC or other store that is hardened against theft, tampering or destruction. Storing of session keys are usually kept in memory by the server and browser, and should be stored in a memory space that is unavailable to other programs. When contained within a cryptographic module, secret and private keys may be stored in plaintext form. These plaintext keys are not be accessible from outside the module. A means is provided to ensure that all keys are associated with the correct entities (i.e., person, group, or process) to which the keys are assigned. Any cryptographic key that it stored in a repository must be protected from disclosure for the life of the key[2]. This includes encryption of the key, restricted access controls, password protection, dispersion and obfuscation among other measures to prevent it from falling into nefarious hands.

D. Delete Keys

A cryptographic module of any web service in an enterprise utilizes an internal capability to zeroize all *plaintext* cryptographic keys and other unprotected critical security parameters within the module when the keys are no longer in use. Zeroization of cryptographic keys and other critical security parameters is not required if the keys and parameters are either encrypted or otherwise physically or logically protected (e.g., contained within an additional embedded FIPS 140-2 cryptographic module)~~.~~

III. Encryption

In cryptography, **encryption** is the process of transforming information (*plaintext*) using an algorithm (*cipher*) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key. The result of the process is *ciphertext*. Encryption can be used to protect data "at rest", such as files on computers and storage devices, or to protect data in transit, for example data being transferred via networks. There have been numerous reports of data in transit being intercepted in recent years [7]. Encryption can protect the confidentiality of messages, but other techniques are still needed to protect the integrity and authenticity of a message; for example, verification of MAC or a digital signature.

Symmetric vs. Asymmetric encryption algorithms

Symmetric encryption algorithms encrypt and decrypt with the same key. Main advantages of symmetric algorithms are its security and high speed. Asymmetric encryption algorithms encrypt and decrypt with different keys. Data is encrypted with a public key, and decrypted with a private key (or vice versa). Generally, symmetric encryption algorithms are much faster to execute than asymmetric ones. In practice they are often used together, so that a public-key algorithm is used to encrypt a randomly generated encryption key, and the random key is used to encrypt the actual message using a symmetric algorithm.

A. *Asymmetric Encryption [12]*

Asymmetric Encryption keys come in pairs. What one key encrypts, only the other can decrypt. Frequently the keys are interchangeable, in the sense that if key A encrypts a message, then B can decrypt it, and if key B encrypts a message, then key A can decrypt it. Asymmetric Encryption is also known as Public Key Cryptography, since entities typically are assigned a matching key pair, and make one public while keeping the other secret. Entities can "sign" messages, and send secret messages by encrypting a message with the recipient's public key. The requisite to successful use of asymmetric encryption is a Key Management system, which implements a Public Key Infrastructure [5b].

Asymmetric Encryption algorithms

There are a number of algorithms in use, for reference; Rivest, Shamir and Adleman - (RSA), Digital Signature Algorithm – (DSA), Pretty Good Privacy – (PGP). Keys that are RSA [6] compatible are generated for the enterprise X.509 certificates.

RSA asymmetric encryption

RSA uses public and private keys that are functions of a pair of large prime numbers based on the difficulty of factoring large integers. The keys used for encryption and decryption in RSA algorithm, are generated using random data. The key used for encryption is a public key. Public keys are stored anywhere publicly accessible. The sender of message encrypts the data using public key, and the receiver decrypts it using his/her own private key.

Combination of Symmetric and Asymmetric Encryption

If we want the benefits of both types of encryption algorithms, the general idea is to create a random symmetric key to encrypt the data, and then encrypt that key asymmetrically. Once the key is asymmetrically encrypted, we add it to the encrypted message. The receiver gets the key, decrypts it with their private key, and uses it to decrypt the message.

B. *Symmetric Encryption*

Symmetric Encryption uses the same secret key to encrypt and decrypt information or there is a simple transform between the two keys. A secret key can be a number, a word, or just a string of random letters. Symmetric algorithms require that both the sender and the receiver know the secret key.

Symmetric-key algorithms [12]

Symmetric-key algorithms can be divided into Stream algorithms (Stream ciphers) and Block algorithms (Block ciphers).

Stream Ciphers

Stream ciphers encrypt the bits of information one at a time - operate on 1 bit (or sometimes 1 byte) of data at a time (encrypt data bit-by-bit). Stream ciphers are faster and smaller to implement than block ciphers, however, they have an important security gap. If the same key stream is used, certain types of attacks may cause the information to be revealed. This document does not specify stream ciphers.

Block Ciphers

Block cipher (method for encrypting data in blocks) is a symmetric cipher which encrypts information by breaking it down into blocks and encrypting data in each block. A block cipher encrypts data in fixed sized blocks (commonly of 64 bits). The most used block ciphers are Triple DES and AES. The enterprise uses AES for block ciphers.

AES/Rijndael encryption

AES stands for Advanced Encryption Standard. AES is a symmetric key encryption technique which replaces the commonly used Data Encryption Standard (DES). It was the result of a worldwide call for submissions of encryption algorithms issued by the US Government's National Institute of Standards and Technology (NIST) in 1997 and completed in 2000. The winning algorithm, Rijndael, was developed by two Belgian cryptologists, Vincent Rijmen and Joan Daemen. [3a]. The AES algorithm uses three key sizes: a 128-, 192-, or 256-bit encryption key. Each encryption key size causes the algorithm to behave slightly differently, so the increasing key sizes not only offer a larger number of bits with which you can scramble the data, but also increase the complexity of the cipher algorithm.

Description of the AES cipher

AES is based on a design principle known as a Substitution permutation network. It is fast in both software and hardware. Unlike its predecessor, DES, AES does not use a Feistel network. AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits, whereas Rijndael can be specified with block and key sizes in any multiple of 32 bits, with a minimum of 128 bits. The blocksize has a maximum of 256 bits, but the keysize has no theoretical maximum. AES operates on a 4×4 matrix of bytes, termed the state (versions of Rijndael with a larger block size have additional columns in the state). The AES cipher is specified as a number of repetitions of transformation rounds that convert the input *plaintext* into the final output of *ciphertext*. Each round consists of several processing steps, including one that depends on the encryption key. The key cipher algorithm process involves matrix transformations [3a]. Hardware for performing these functions exists, and may be used. Intel cores have added instruction sets to speed the algorithms.

Data Encryption Standard (DES) [13]

Data Encryption Standard (DES) is a symmetric block cipher developed by IBM. The algorithm uses a 56-bit key to encipher/decipher a 64-bit block of data. The key is always presented as a 64-bit block, every 8th bit of which is ignored. The algorithm is best suited to implementation in hardware. DES is the most widely used symmetric algorithm in the world, despite claims that the key length is too short. Ever since DES was first announced, controversy has raged about whether 56 bits is long enough to guarantee security. NIST recommended that DES should be replaced by Triple DES, a modified version employing 112- or 168-bit keys. DES's versatility also was limited because it worked only in hardware, and the explosion of the Internet and e-commerce led to much greater use and versatility of software than could have been anticipated by DES's designers. As DES's vulnerabilities became apparent, NIST opened an international competition in 1997 to find a permanent replacement for DES (see AES cipher above).

Triple DES

Triple DES is a variation of DES. It uses a 64-bit key consisting of 56 effective key bits and 8 parity bits. The size of the block for Triple-DES is 8 bytes. Triple-DES encrypts the data in 8-byte chunks. The idea behind Triple DES is to improve the security of DES by applying DES encryption three times using three different keys.

Description of the Triple DES cipher [13]

Those who consider the exhaustive key-search attack to be a real possibility can make the problem more difficult for an adversary by using double or triple length keys. Use of multiple length keys leads us to the Triple-DES algorithm, in which DES is applied three times. If we consider a triple length key to consist of three 56-bit keys K1, K2, K3 then encryption is as follows:

1. Encrypt ***plaintext*** with K1
2. Decrypt ***result in 1*** with K2
3. Encrypt ***result in 2*** with K3 to provide ***ciphertext***

Setting K3 equal to K1 in these processes gives us a double length key K1, K2. Setting K1, K2 and K3 all equal to K has the same effect as using a single-length (56-bit key). Thus it is possible for a system using triple-DES to be compatible with a system using single-DES. The key cipher algorithm process involves matrix transformations [3c].

IV. Decryption

Decryption is the complement of encryption. In cryptography, **decryption** is the process of transforming information (*ciphertext* using an algorithm (*cipher*) to make it readable to an entity that possesses special knowledge (key). The result of the process is **decrypted** information (in cryptography, referred to as *plaintext*)).

A. Asymmetric Decryption

Since we have specified that asymmetric encryption and decryption is by PKI properties of the enterprise certificate for active entities we use the RSA algorithm by inverting the encryption process and using the complementary key.

B. Symmetric Decryption

For symmetric encryption we have limited the possible algorithms to AES and Triple DES.

The AES decryption algorithm is applied as follows:

Recall: The AES cipher is specified as a number of repetitions of transformation rounds that convert the input *plaintext* into the final output of *ciphertext*. Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform *ciphertext* back into the original *plaintext* using the same encryption key.

The Triple DES decryption algorithm is applied as follows:

Decryption is the reverse of the encryption process (see Description of the Triple DES process):

1. Decrypt ***ciphertext*** with K3
2. Encrypt ***result of 1*** with K2
3. Decrypt ***result of 2*** with K1 to provide ***plaintext***

Setting K3 equal to K1 in these processes gives us a double length key K1, K2. Setting K1, K2 and K3 all equal to K has the same effect as using a single-length (56-bit key). Thus it is possible for a system using triple-DES to be compatible with a system using single-DES. The key cipher algorithm process involves matrix transformations [3c].

V. Hash Function

A **hash function** is any algorithm or subroutine that maps *plaintext* to a value, called keys. For example, a value can serve as an index to list of *plaintext* items. Hash functions are mostly used to accelerate table lookup or data comparison tasks such as finding items in a database, detecting duplicated or similar records in a large file, finding similar stretches in DNA sequences, and so on. In cryptographic services, a hash is used to nearly uniquely identify the content of a *plaintext.* Some hash functions may map two or more *plaintexts* to the same hash value, causing a collision. Cracking a hash code would be devising a set of rules by which one change can be compensated with other changes to make the hash the same. Such hash functions try to map the *plaintext* to the hash values as evenly as possible because, as Hash Tables fill up, collisions become more frequent.

Hash function algorithms

For most types of hashing functions the choice of the function depends strongly on the nature of the input data, and their probability distribution in the intended application, in this case we use *plaintext* which consists of SAML tokens, messages in SOAP envelopes, log records, etc. The hash is part of the almost unique identifier bound to a signature[15] (see VI below)..

Hashing with cryptographic hash functions

Some cryptographic hash functions, such as SHA-3/SHA-512, have stronger uniformity guarantees than checksums or fingerprints, and thus can provide very good general-purpose hashing functions. A cryptographic hash function is a deterministic procedure that takes an arbitrary block of data (*plaintext*) and returns a fixed-size bit string; the **(cryptographic) hash value,** such that an accidental or intentional change to the data changes the hash value. The data to be encoded (*plaintext*) is often called the "message," and the hash value is sometimes called the **message digest**, **digest** or **hash.** The ideal cryptographic hash function has four main or significant properties; it is easy to compute, but it is unlikely to: generate a message that has a given hash, change a message and not change the hash, or find different messages with the same hash. There are a large number of cryptographic hash algorithms but for the enterprise we restrict usage to two such constructs:

- MD-5 Message Digest 5
- SHA-512 Secure Hash Algorithm 512 bits

MD-5

The MD5 Message-Digest Algorithm is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value. MD5 is commonly used to check data integrity. In an attack on MD5 published in December 2008, a group of researchers used this technique to fake SSL certificate validity. US-CERT has advised that MD5 "should be considered cryptographically broken and unsuitable for further use" and most U.S. enterprise applications now require the SHA-2 family of hash functions. For this reason, MD-5 is not to be used in standalone operations but may be used in HashMAC (HMAC) construction for TLS.

SHA-3 defined SHA-512

In cryptography the Secure Hash Algorithm (SHA), **SHA-2** is a set of cryptographic hash functions (**SHA-224, SHA-256, SHA-384, and SHA-512**) designed by the National Security Agency (NSA) and published in 2001 by the NIST as a U.S. Federal Information Processing Standard. SHA stands for **Secure Hash Algorithm**. SHA-2 includes a significant number of changes from its predecessor, SHA-1. In October 2012, the National Institute of Standards and Technology (NIST) chose a new algorithm (Keccak) as the SHA-3 standard [8]. A notable problem in moving from SHA-1 to SHA-2 is that they both used the same algorithmic approach (Merkle-Damgard), to process message text. This means that a full attack on SHA-1 becomes a potential threat for SHA-2. While no successful attacks against a full-round SHA-2 have been announced, there is no doubt that attack mechanisms are being developed in private. For the enterprise the goal is that SHA-3/SHA-512 be used for all signature data. SAHA-2 defined SHA-256 or -224 are currently considered adequate. The SHA-3 standard has yet to be published even though the algorithm has been chosen. However, because of the compromises of other hash algorithms, and the time required to implement new models, a transition to SHA-3 defined SHA-512 should be undertaken. It may be combined with MD-5 for HMAC applications in TLS.

VI. Signatures

A digital signature is an information element that can be added to a document (passive entity) that can be used to authenticate the identity of the generator or the signer of a document, and ensure that the original content of the document that has been sent is unchanged. Digital signatures are easily transportable, cannot be imitated by someone else. The ability to ensure that the original signed message arrived means that the sender cannot easily repudiate it later. The digital signature has specific content elements that include an encrypted hash of the document, hash details, a time stamp, and the X.509 certificate of the signer. An important feature of each signature is the ability to validate the signature.

The attachment of the X.509 of the signer which must be verified and validated through a number of steps:

a. The certificate is issued by a trusted certificate authority
b. The certificate date is valid and the distinguished name of the signer matches the certificate.
c. The certificate is not revoked

The hash must be validated by decrypting with the public key of the signer, and comparing it to an independently computed hash. For the enterprise we consider three basic signatures:

- XML Signatures
- S/MIME Signatures Secure/Multipurpose Internet Mail Extensions (S/MIME)
- E-Content Signatures

A. *XML Signature*

XML signatures are digital signatures designed for use in XML transactions. They are defined by a standard schema for capturing the result of a digital signature operation applied to arbitrary (but often XML) data. Like non-XML-aware digital signatures (e.g., PKCS), XML signatures add authentication, data integrity, and support for non-repudiation to the data that they sign. However, unlike non-XML digital signature standards, XML signature has been designed to both account for and take advantage of the Internet and XML. A fundamental feature of XML Signature is the ability to sign only specific portions of the XML tree rather than the complete document. The entire return may be encrypted for security [1a].

B. *S/MIME Signature*

S/MIME (Secure/Multipurpose Internet Mail Extensions) provides a consistent way to send and receive secure MIME data. Based on the popular Internet MIME standard, S/MIME provides the following cryptographic security services for electronic messaging applications: authentication, message integrity and non-repudiation of origin (using digital signatures), and data confidentiality (using encryption). S/MIME can be used by traditional mail user agents (MUAs) to add cryptographic security services to mail that is sent, and to interpret cryptographic security services in mail that is received. However, S/MIME is not restricted to mail; it can be used with any transport mechanism that transports MIME data, such as HTTP. S/MIME [5j] provides one format for enveloped-only data, several formats for signed-only data, and several formats for signed and enveloped data.

C. *E-Content Signature*

The Digital Signature Standard (DSS) (FIPS 186-3) developed by NIST is used for general E-Content signing. This Standard specifies algorithms for applications requiring a digital signature, rather than a written signature. A digital signature is represented in a computer as a string of bits. A digital signature is computed using a set of rules and a set of parameters that allow the identity of the signatory and the integrity of the data to be verified. Digital signatures may be generated on both stored and transmitted data. Signature generation uses a private key to generate a digital signature; signature verification uses a public key that corresponds to, but is not the same as, the private key. Each signatory possesses a private and public key pair. Public keys may be known by the public; private keys are kept secret. Anyone can verify the signature by employing the signatory's public key. Only the user that possesses the private key can perform signature generation. A hash function is used in the signature generation process to obtain a condensed version of the data to be signed; the condensed version of the data is often called a message digest. The message digest is input to the digital signature algorithm to generate the digital signature. The hash functions to be used are specified in the Secure Hash Standard (SHS) [3f]. FIPS approved digital signature algorithms are used with SHA-512. The digital signature is provided to the intended verifier along with the signed data. The verifying entity verifies the signature by using the claimed signatory's public key and the same hash function that was used to generate the signature. Similar procedures may be used to generate and verify signatures for both stored and transmitted data [3f].

VII. Summary

We have reviewed a set of cryptographic processes for confidentiality, integrity and authentication. Web-based architectures require cryptographic services at the application layer. The high assurance architecture demands strong cryptographic services for data both in transit and at rest. The strength of cryptographic functions in today's environment is beyond previous recommendations due to the improvements in computing and the adaptability of the threat. The cryptographic process is part of a comprehensive enterprise architecture for high assurance that is web-service based and driven by commercial standards. Portions of this architecture are described in references [9 – 11].

References

[1]. World Wide Web Consortium (W3C):
a. XML Encryption Syntax and processing, 10 Dec 2002.
b. XML Signature Syntax and Processing,10 Jun 2008.
c. Canonical XML Ver. 1, Mar 2001.
d. XML-Exclusive Canonicalization Ver. 1, Jul, 2002.

[2]. OASIS open set of Standards:
a. "WS-Security Specification 1.1" OASIS, Nov 2006
b. "WS-Trust Specification 1.4". OASIS, Feb 2009
c. "WS-ReliableMessaging Specification 1.1", OASIS, 2004
d. "WS-SecureConversation Specification 1.4", OASIS, 2009

[3]. National Institute of Standards, Gaithersburg, Md:
a. FIPS 197, Advanced Encryption Standard (AES), Nov 2001.
b. Publication 800-38A-C, December 2001- May 2004.
c. FIPS 800-67, Ver. 1.2 , Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Jan 2012
d. FIPS PUB 140-2, Security Requirements for Cryptographic Modules, May 25, 2001
e. FIPS PUB 186-3, Digital Signature Standard, Jun, 2009
f. FIPS PUB 180-3. Secure Hash Standard. Aug 2002

[4]. Common Criteria for Information Technology Security Evaluation, Version 3.1, revision 3, July 2009.

[5]. Internet Engineering Task Force (IETF) Standards:
a. RFC 5246: The Transport Layer Security Protocol 1.2, 2008.
b. RFC 2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile, Jan 1999.
c. RFC 6151: Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms, Mar 2011.
d. RFC 0959, STD 9 File Transfer Protocol, J. Postel, J. Reynolds, Oct 1985.
e. RFC 0791, STD 5, Internet Protocol, J. Postel, Sep1981, and subsequent RFCs 791/950/919/922/792/1112.
f. RFC 4510 Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map, June 2006
g. RFC 5751 Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification, July 2010.
h. RFC 2829, Authentication Methods for LDAP, May 2000.

[6]. PKCS #1: RSA Cryptography Standard, http://www.rsa.com/rsalabs/node.asp?id=2125

[7]. Miller, Sandra Kay, Fiber Optic Networks Vulnerable to Attack, Information Security Magazine, November 15, 2006,

[8]. José R.C. Cruz, Dr. Dobb's, The World of Software Development, "Keccak: The New SHA-3 Encryption Standard", May 2013, http://www.drdobbs.com/security/keccak-the-new-sha-3-encryption-standard/240154037 see http://keccak.noekeon.org/

[9]. William R. Simpson, Coimbatore Chandersekaran and Andrew Trice, The 1st Int'l Multi-Conf on Eng. and Tech Innovation: IMET2008, "Cross-Domain Solutions in an Era of Information Sharing", Vol. I, pp.313-318, Orlando, FL., June 2008.

[10]. Coimbatore Chandersekaran and William R. Simpson, W3C Workshop on Security Models for Device APIs, "The Case for Bi-lateral End-to-End Strong Authentication", 4 pp., London, England, December 2008.

[11]. William R. Simpson and Coimbatore Chandersekaran, The 2nd Int'l Multi-Conf. on Eng. and Tech Innovation: IMETI2009, Vol. I, pp. 300-305, "Information Sharing and Federation", Orlando, FL., July 2009.

[12]. Some material adapted from InfoBlox, http://www.encryptionanddecryption.com/encryption/asymmetric_encryption.html.

[13]. Taken in part from Cryptography World, http://www.cryptographyworld.com/des.htm

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)
00-10-2013

2. REPORT TYPE
Non-Standard

3. DATES COVERED (From – To)

4. TITLE AND SUBTITLE
Cryptography for a High-Assurance Web-Based Enterprise

5a. CONTRACT NUMBER
DASW01-04-C-0003

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBERS

6. AUTHOR(S)
William R. Simpson, Coimbatore Chandersekaran

5d. PROJECT NUMBER

5e. TASK NUMBER
BC-5-2283

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESSES
Institute for Defense Analyses
4850 Mark Center Drive
Alexandria, VA 22311-1882

8. PERFORMING ORGANIZATION REPORT NUMBER
NS D-5090
H 2013-001837

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Frank P. Konieczny
USAF HQ USAF SAF/CIO A6

10. SPONSOR'S / MONITOR'S ACRONYM
SAF A6 / CTO

11. SPONSOR'S / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for Public Release

13. SUPPLEMENTARY NOTES
Project Leader: William R. Simpson

14. ABSTRACT

Each web service and each infrastructure service has a need for symmetric and asymmetric encryption, as well as signature processing and other cryptographic processes. This profile does not include electrical requirements for Multi-Level Systems (MLS). A number of specialized cryptographic functions have been developed for hardware and network operations. Their use is appropriate for network level operations. For purposes of this paper, the discussion is limited to NSA Type 3 for IP enabled communications and similar algorithms. Cryptography is used by most of the services in an enterprise. Asymmetric encryption is performed in suitably security hardened stores and symmetric encryption is performed in most bi-lateral operations. Signatures for integrity and trust use are pervasive. Key management is required throughout the enterprise. The crypto services may be used through all of the Open Systems Interconnection (OSI) model layers, however, this document concentrates on layers 4 and above.

15. SUBJECT TERMS
Public Key Infrastructure; Electronic Signature; Web services; Transport Layer Security; Cryptographic Hash.

16. SECURITY CLASSIFICATION OF:

a. REPORT
Unclassified

b. ABSTRACT
Unclassified

c. THIS PAGE
Unclassified

17. LIMITATION OF ABSTRACT
Unlimited

18. NUMBER OF PAGES
6

19a. NAME OF RESPONSIBLE PERSON
Frank P. Konieczny

19b. TELEPHONE NUMBER (Include Area Code)
(703) 697-1308